PCI-L122-DSF

PCI-L122-DSF User Manual

Version: V1.1 2014A08

To properly use the product, read this manual thoroughly is necessary.

Part No.: 81-02L122F-011

Revision History

Date	Revision	Description
2012/6/30	1.0	Document creation.
2014/4/8	1.1	Reformat API introductions and append some functions.

© Copyright 2010 TPM

The product, including the product itself, the accessories, the software, the manual and the software description in it, without the permission of TPM Inc. (“TPM”), is not allowed to be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means, except the documentation kept by the purchaser for backup purposes.
The names of products and corporations appearing in this manual may or may not be registered trademarks, and may or may not have copyrights of their respective companies. These names should be used only for identification or explanation, and to the owners’ benefit, should not be infringed without any intention.
The product’s name and version number are both printed on the product itself. Released manual visions for each product design are represented by the digit before and after the period of the manual vision number. Manual updates are represented by the third digit in the manual vision number.

Trademark

- **MS-DOS** and **Windows 95/98/NT/2000/XP**, **Visual Studio**, **Visual C++**, **Visual BASIC** are registered trademarks of Microsoft.
- **BCB (Borland C++ Builder)** is registered trademark of Borland.
- **MULTIPROG** is registered trademark of KW software.
- Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Electrical safely

- To prevent electrical shock hazard, disconnect the power cable from the electrical outlet before relocating the system.
- When adding or removing devices to or from the system, ensure that the power cables for the devices are unplugged before the signal cables are connected. Disconnect all power cables from the existing system before you add a device.
- Before connecting or removing signal cables from motherboard, ensure that all power cables are unplugged.
- Seek professional assistance before using an adapter or extension card. These devices could interrupt the grounding circuit.
- Make sure that your power supply is set to the voltage available in your area.
- If the power supply is broken, contact a qualified service technician or your retailer.

Operational safely

- Please carefully read all the manuals that came with the package, before installing the new device.
- Before use ensure all cables are correctly connected and the power cables are not damaged. If you detect and damage, contact the dealer immediately.
- To avoid short circuits, keep paper clips, screws, and staples away from connectors, slots, sockets and circuitry.
- Avoid dust, humidity, and temperature extremes. Do not place the product in any area where it may become wet.
- If you encounter technical problems with the product, contact a qualified service technician or the dealer.

Contents

CONTENTS **5**

1. INTRODUCTION **7**

1.1. MOTIONNET DEVICES 7
1.1.1. Placement of the Board 8
1.1.2. Motionnet Extension Connector 8
1.1.3. Digital I/O Connector 9
1.1.4. Card Number Switch (RSW1) 11

2. MOTIONNET INTRODUCTION **12**

2.1. WHAT IS MOTIONNET? 12
2.2. MOTIONNET FUNCTIONS 12
2.3. ADVANTAGE OF MOTIONNET 13
2.4. MOTIONNET PRODUCT FAMILY 15

3. PROJECT ENCRYPTION **16**

3.1. BENEFITS 16
3.2. AES BRIEF INTRODUCTION 16
3.3. FUNCTIONAL ARCHITECTURE 16

4. FUNCTION REFERENCE **18**

4.1. MOTIONNET INITIALIZATION 18
4.1.1. Hardware Initialization 18
4.1.2. Library Initialization 18
4.1.3. Motionnet Master 19
4.2. DATA DEFINITION 21
4.3. PCI-L122-DSF API LIST 22
4.3.1. _l122_dsf_open 23
4.3.2. _l122_dsf_close 24
4.3.3. _l122_dsf_check_switch_card_num 25
4.3.4. _l122_dsf_get_start_ring_num 26
4.3.5. _l122_dsf_set_ring_config 27
4.3.6. _l122_dsf_read_local_input 28
4.3.7. _l122_dsf_read_local_output 29
4.3.8. _l122_dsf_write_local_output 30
4.3.9. _l122_dsf_read_fram_byte 31
4.3.10. _l122_dsf_read_fram_word 32
4.3.11. _l122_dsf_read_fram_dword 33
4.3.12. _l122_dsf_write_fram_byte 34
4.3.13. _l122_dsf_write_fram_word 35

4.3.14. _l122_dsf_write_fram_dword 36
4.3.15. _l122_dsf_get_secure_id 37
4.3.16. _l122_dsf_gen_aes_key 38
4.3.17. _l122_dsf_check_aes_key 39

5. APPENDIX 40

5.1. DEFINITION OF ERROR CODES 40

1. Introduction

1.1. Motionnet Devices

PCI-L122-DSF master control card is designed for users who need to quickly and easily develop applications such as motion control and controls of I/O. PCI-L122-DSF is a standard PCI master control card with high speed real-time network extension called Motionnet. Motionnet is a new series of communicating system designed for versatile automation applications.

PCI-L122-DSF is designed following the Standard PCI form factor. This master can connect up to 64x2=128 slave modules. There are 3 categories of the slave modules, for motion control, digital I/O and miscellaneous functions.

For motion control slave modules, there are 3 types: 1-axis, 2-axes or 4-axes. For digital I/O slave modules, there are 3 types: 32-IN, 32-OUT, and 16-IN & 16-OUT. For miscellaneous slave modules, there are AD/DA models, counter modes or the other customized type models.

The more detailed introduction of PCI-L122-DSF is in the following subsection. The functional block is illustrated in Figure 1-1.

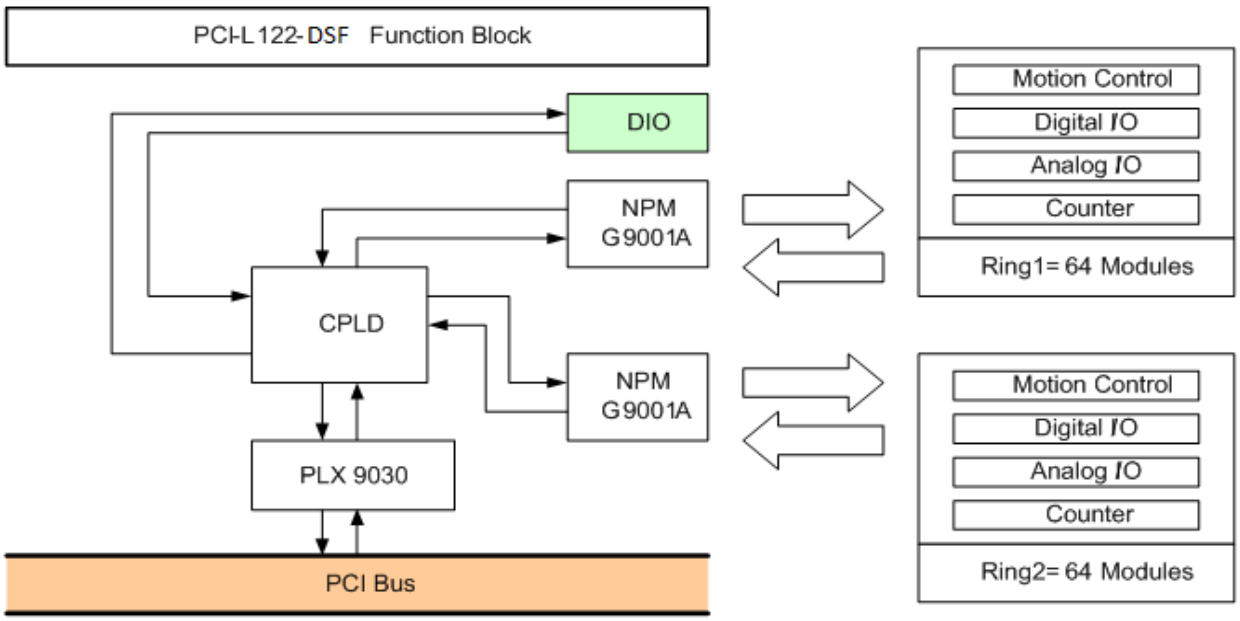

Figure 1-1: PCI-L122-DIO functional block

1.1.1. Placement of the Board

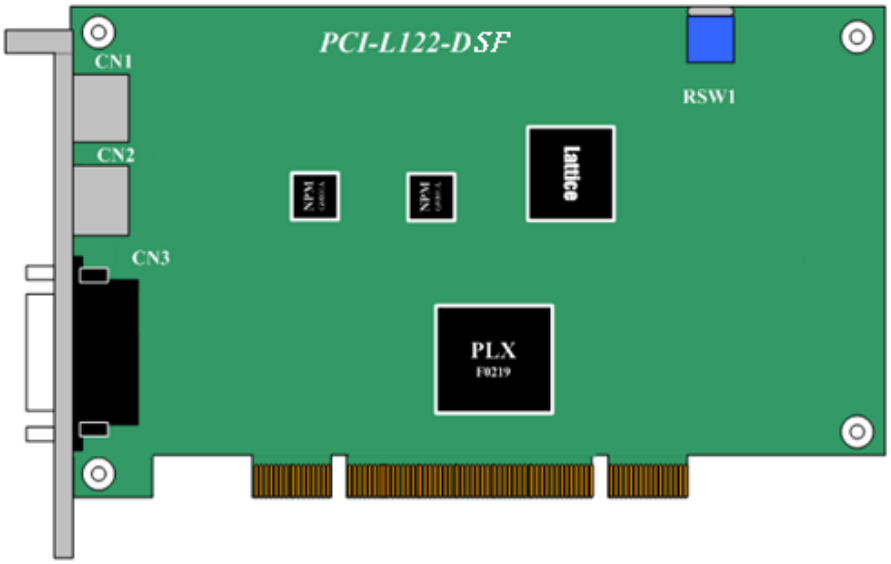

Figure 1-2: placement of PCI-L122-DSF control board

1.1.2. Motionnet Extension Connector

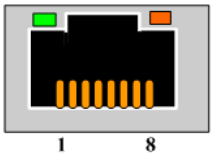

Figure 1-3: Motionnet extension connector Ring0/Ring1 for PCI-L122-DIO

Pin	Label	Description
1	FG	Filed Ground
2	FG	Filed Ground
3	RS485+	High Speed RS-485 protocol
4	FG	Filed Ground
5	FG	Filed Ground
6	RS485-	High Speed RS-485 protocol
7	FG	Filed Ground
8	FG	Filed Ground

1.1.3. Digital I/O Connector

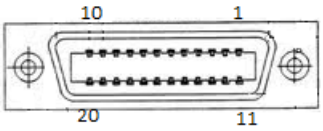

Figure 1-4: digital I/O connector

Pin	Label	Description	Pin	Label	Description
1	EXT_IN0	GPIO Input 0	11	EXT_OUT0	GPIO Output 0
2	EXT_IN1	GPIO Input 1	12	EXT_OUT1	GPIO Output 1
3	EXT_IN2	GPIO Input 2	13	EXT_OUT2	GPIO Output 2
4	EXT_IN3	GPIO Input 3	14	EXT_OUT3	GPIO Output 3
5	EXT_IN4	GPIO Input 4	15	EXT_OUT4	GPIO Output 4
6	EXT_IN5	GPIO Input 5	16	EXT_OUT5	GPIO Output 5
7	EXT_IN6	GPIO Input 6	17	EXT_OUT6	GPIO Output 6
8	EXT_IN7	GPIO Input 7	18	EXT_OUT7	GPIO Output 7
9	DI_COM	GPIO COM	19	Reserved	NC
10	EGND	GPIO GND	20	E24V	External 24V DC

1.1.3.1 GPIO Input of Connection for NPN-Sink Mode

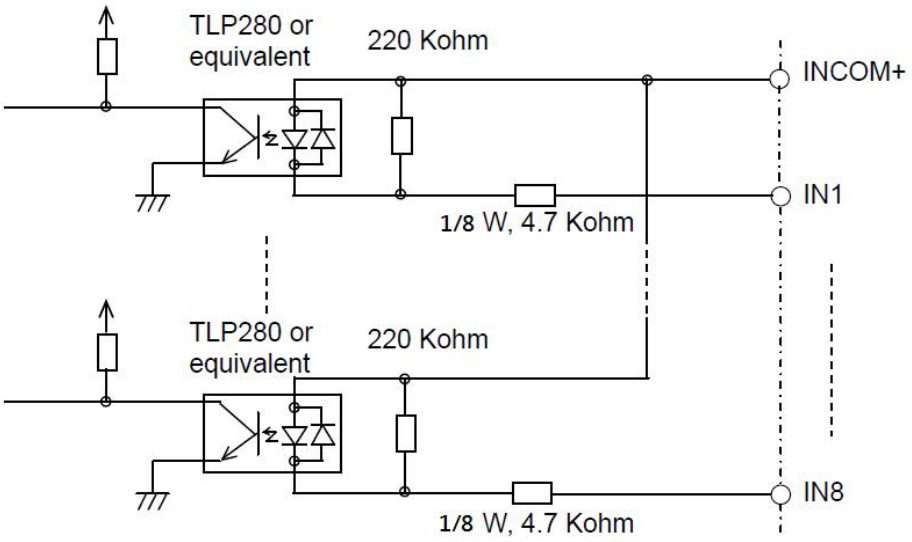

Figure 1-5: GPIO input of connection for NPN-Sink mode

1.1.3.2 GPIO Input of Connection for PNP-Source Mode

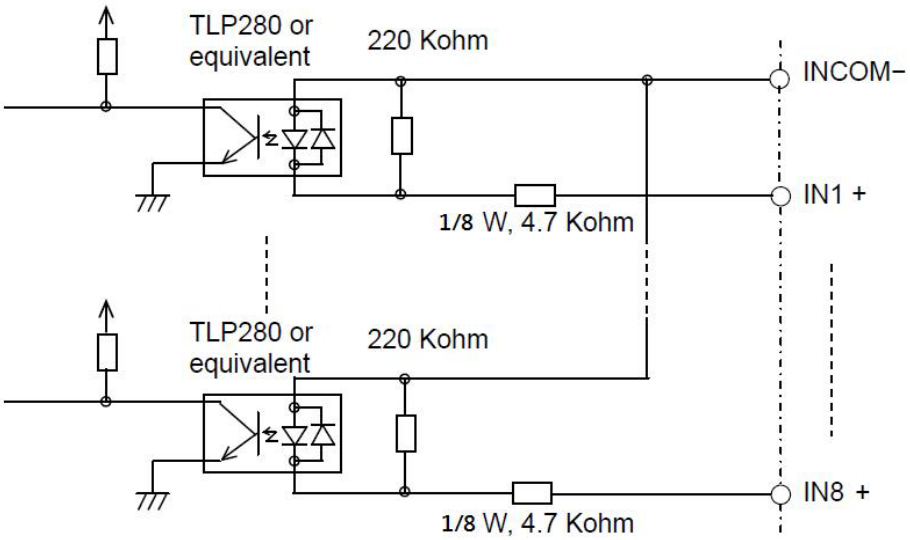

Figure 1-6: GPIO Input of connection for PNP-Source mode

1.1.3.3 GPIO Output of Connection for Transistor

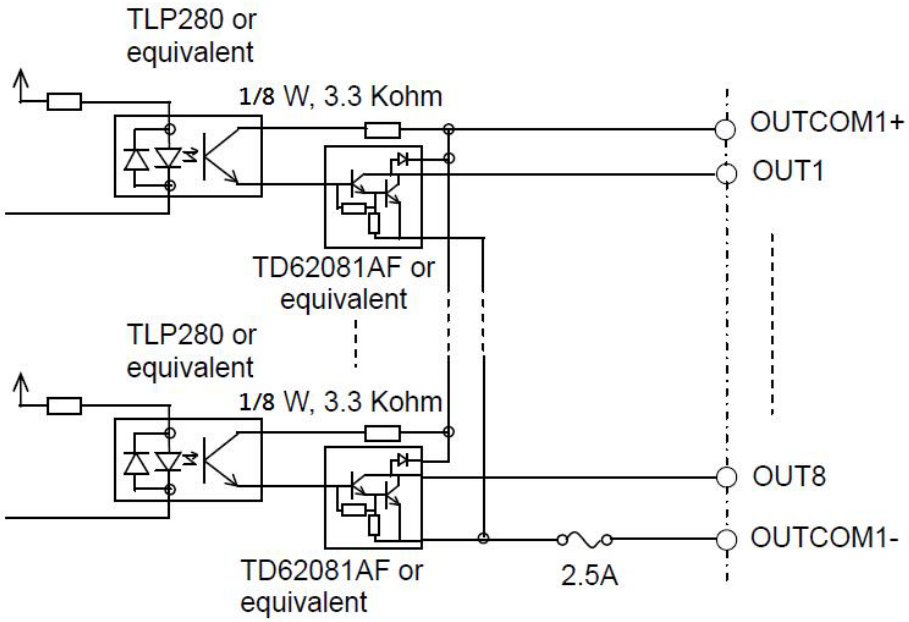

Figure 1-7: GPIO output of connection for transistor

1.1.4. Card Number Switch (RSW1)

If there are 2 PCI-L122-DSF cards installed in the system, it is necessary to specify the ID for each card respectively by configuring the DIP rotary switch as shown below.

Figure 1-8: DIP rotary switch on PCI-L122-DSF

Label	Description
#0 ~ #7	Card #0 to card #7
#8 ~ #F	Reserved

2. Motionnet Introduction

2.1. What Is Motionnet?

Motionnet is a super high-speed serial communication system. The G9000 devices provide input/output control, motor control, CPU emulation and message communication with high speed serial communications (up to 20Mbps) all of which are required by current Factory Automation techniques. Motionnet always transfers 4 bytes of data in 15.1μsec using cyclic communication to control input and output. While this data is being transferred, it can communicate a maximum of 256 bytes, such as motor control data, and the LSI controls the data transmission using interrupts. Communication times can be calculated using formulas, allowing users to see that Motionnet guarantees the real-time oriented support needed by FA industries.

2.2. Motionnet Functions

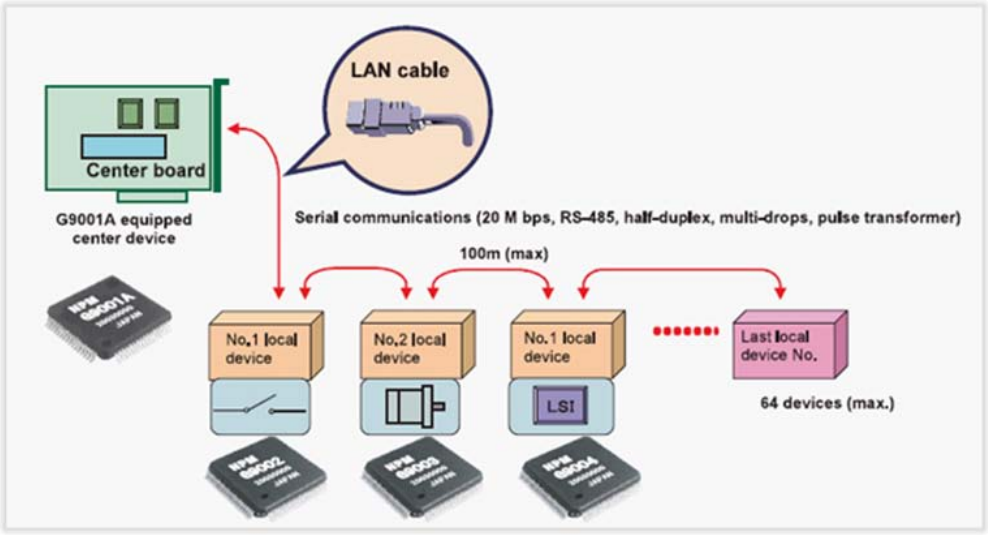

Figure 2-1: Motionnet system architecture

- Provides a communication protocol based on the RS485 standard.
- Can communicate variable length of data from 1 to 128 words (when a 16-bit CPU is used)
- An LSI center device (G9001) controls the bus.
- I/O wiring can be greatly reduced by using a G9002 I/O device.
- Motor control wiring can be reduced by using a G9003 PCL.
- Using a G9004 CPU emulation device reduces the wiring for general devices connected to a CPU. Data can be exchanged between CPUs by changing the G9004 mode.
- New devices can be added to the system on the fly.
- Systems can be isolated using pulse transformers.
- Transfer speed up to 20 Mbps.
- Maximum 64 slave devices for each serial line on a master device. Input/output control of up to 256 ports

(2048 points), motion control of up to 64 axes, and LSI control of up to 128 devices.

- Input/output and status communication time for each device when inputting/outputting and reading status data for each device, the system automatically refreshes the center device RAM each communication cycle. (Cyclic communication: 15.1 µsec./local device) When 32 local devices are connected (1024 points of input/output): 0.49 msec. When 64 local devices are connected (2048 points of input/output): 0.97 msec.
- Data communication time cyclic communication can be interrupted with a command from the CPU. Data communication time: 19.3 µsec. to send or receive 3 bytes (e.g. when writing feed amount data to the G9003). Data communication time: 169.3 µsec. to send or receive 256 bytes.
- Serial communication connection cable. Multi-drop connections using LAN cables or dedicated cables. Total cable length of one line: 100 m (20 Mbps/32 local boards) (10 Mbps/64 local boards). Cable length between local boards: 0.6 m or longer.

2.3. Advantage of Motionnet

- It is possible to connect from center to terminal controller parts by one cable.

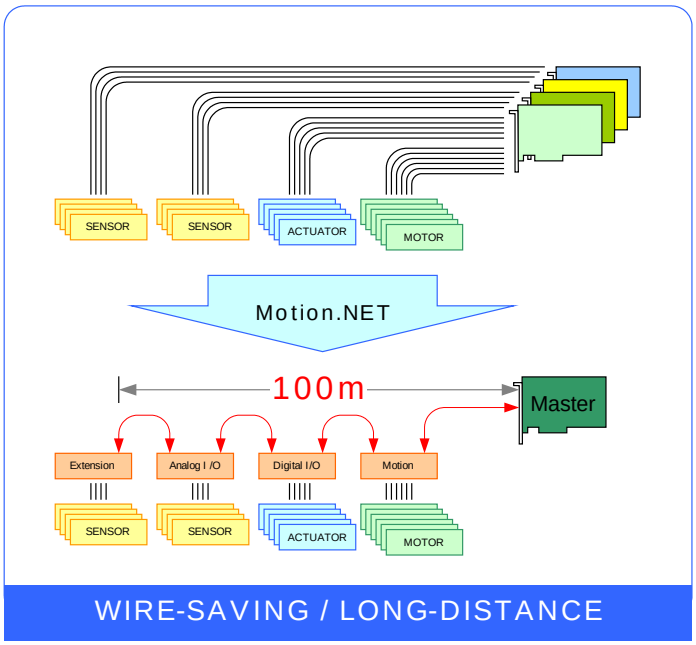

Figure 2-2: wire-saving and long-distance support

- In cyclic communication, a communication cycle is as follows when a 20 Mbps speed is selected.

Number of local devices	Communication cycle	Remarks
8	0.12 ms	If all of the local devices connected are I/O devices, 256 input/output points can be used.
16	0.24 ms	If all of the local devices connected are I/O devices, 512 input/output points can be used.
32	0.49 ms	If all of the local devices connected are I/O devices, 1024 input/output points can be used.
64	0.97 ms	If all of the local devices connected are I/O devices, 2048 input/output points can be used.

If a different number of local devices are connected, or when the communication cycle is interrupted by data communications, refer to the calculation formulas in the user's manual to calculate the time latency.

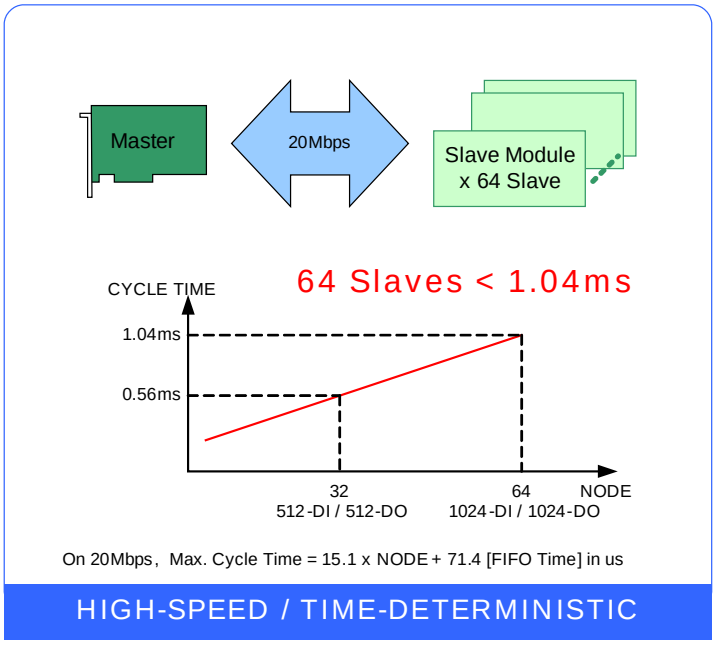

Figure 2-3: high-speed and time deterministic support

2.4. Motionnet Product Family

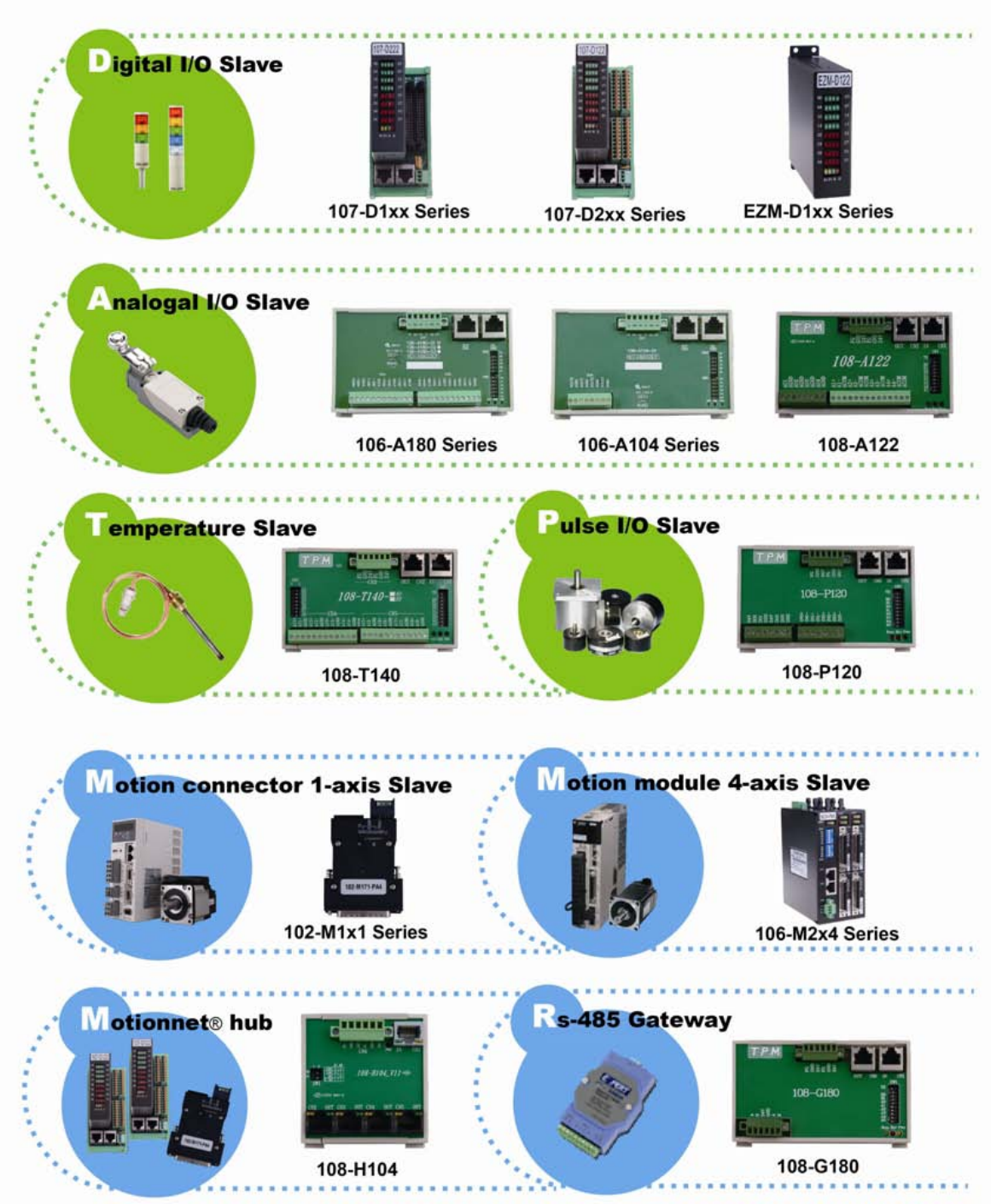

Figure 2-4: Motionnet product family

3. Project Encryption

This chapter is intended to give a brief overview of the project encryption for PCI-L122-DSF. The following section will give background information that is necessary to fully understand the functions and how to encrypt a project of the system.

3.1. Benefits

TPM is a development system provider with PCI-L122-DSF as one of the products. System integrators could adapt PCI-L122-DSF as the base system to develop applications for their customers. However, customers could find the top source vendor which is TPM and perhaps, the worst case, clone the storage in the system and purchase extra systems from TPM directly. In case of customers bypass the original system provider, which would cut down benefits for the system integrators cooperating with TPM, PCI-L122-DSF introduces a technology called project encryption. Through project encryption, the system integrators can lockup certain functionalities or set timers to constrain the system running time. Only the authorized products can be working properly. The authorization is hold by the one and only one system integrator.
With the project encryption technology, the system integrators cooperating with TPM and TPM will be tightly coupled cooperating relationship instead of vicious competition to make a win-win partnership.

3.2. AES Brief Introduction

This standard specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. Rijndael was designed to handle additional block sizes and key lengths. However they are not adopted in this standard. Throughout the remainder of this standard, the algorithm specified herein will be referred to as “the AES algorithm.” The algorithm may be used with the three different key lengths indicated above, and therefore these different “flavors” may be referred to as “AES-128”, “AES-192”, and “AES-256”.
Since the AES encryption/decryption algorithm is not the main function of PCI-L122-DSF, the detailed introduction is not introduced in this manual. Please refer to Wikipedia for more information.

3.3. Functional Architecture

Each PCI-L122-DSF equipped an identification chip with unique serial number. The unique serial number plays the role as the content for AES algorithm. We call the unique serial number the hardware id from now on. Another key held by the system integrator is called the SI key, used for encryption/decryption for the AES algorithm to make the registration key. The illustration of the making of the registration is as the following figure.

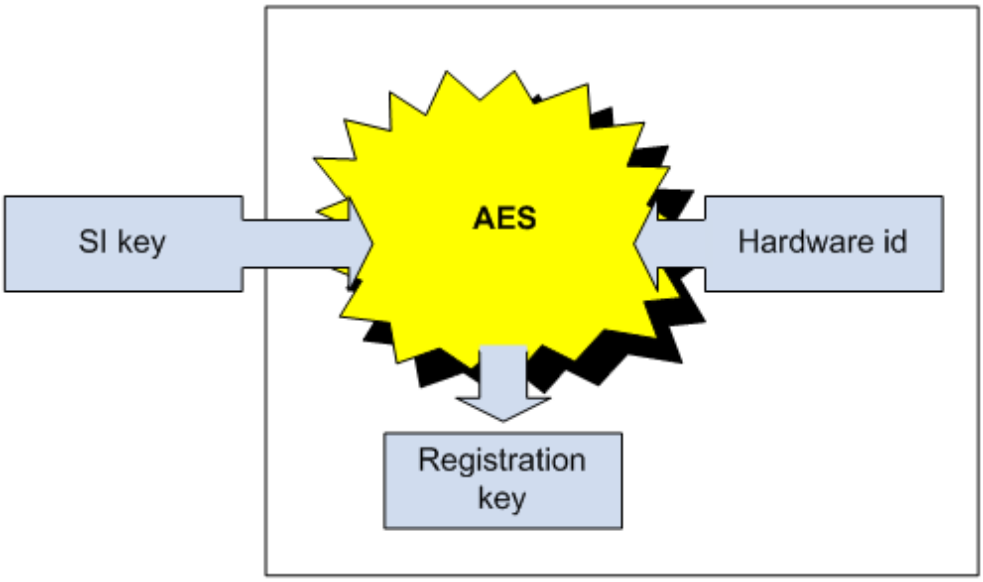

Figure 3-1: generation of the registration key

From the above figure, the hardware id is obtainable within PCI-L122-DSF, taken as the content for AES algorithm. The SI key, hard coded by the system integrator, is the key to calculate the output value, the registration key. The registration key is like the activation code to activate the full functionalities of the PCI-L122-DSF and it is not invertible. Even though the whole data including the registration key could be cloned, the end user or the system integrator's customer cannot obtain the SI key. If an end user wants to buy a replacement from other resources instead of the designated system integrator is not doable since the hardware id would be different with the original one. Therefore, the calculated registration key with the original SI key and different hardware id as content would not match so that the specific functionalities could not be working or the system could only run for certain time period.

Once the system integrator gets PCI-L122-DSF, a very important thing need to do is to assign a set of SI key with a byte array of length 8 to it. With this SI key, the system integrator could generate the AES key (byte array of length 16) using the provided function application interface _L122DSF_gen_aes_key. This function will be introduced later. The generated AES key should be given to end users as the activation code when the user brings up the system at first time. A PCI-L122-DSF should check the validity and set the corresponding flag in the FRAM right after the system initialization using function _L122DSF_check_aes_key, _L122DSF_fram_write_word. Hereafter when the system is brought up, it reads the flag from FRAM. If the read value matches the pre-defined value, the system bypasses the check AES key procedure and operates normally as the system integrator designed. If the values from the designated address in FRAM do not match, the system could only run certain time or some advanced functions are locked. Only with the correct activation code can bring the system up with full functionalities and unlimited running time.

4. Function Reference

Motionnet is a low cost, digital-serial control interface for communications between host algorithm and axis-controllers, I/O devices and other function devices. Most physical or data layer tasks are completed by the ASIC hardware together with user-friendly software under Windows operating system. This chapter describes the operating principle and application interfaces of the Motionnet. This chapter is helpful to users want to know more details about the operational principle of the motion control card.

4.1. Motionnet Initialization

The system initialization is divided into two parts: hardware initialization and library initialization. The hardware and system layer is as shown in the following figure.

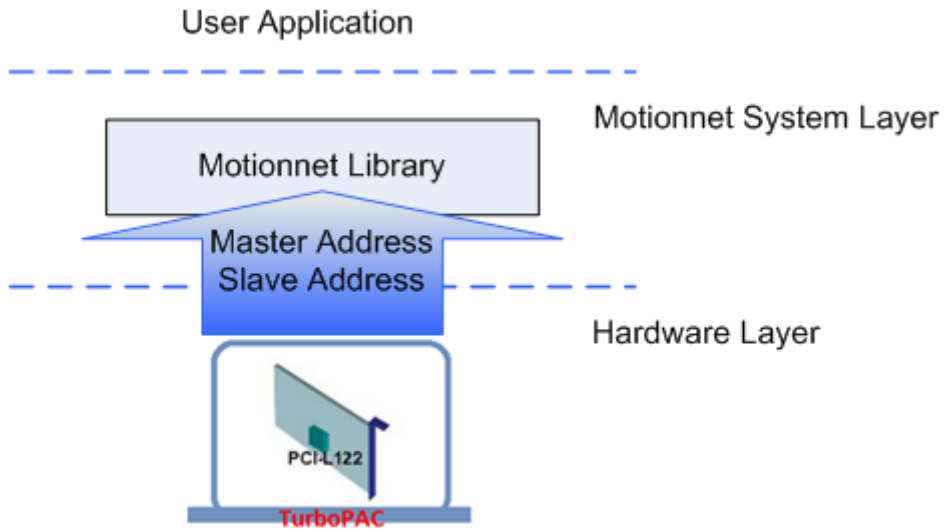

Figure 4-1: Motionnet system and hardware layer

4.1.1. Hardware Initialization

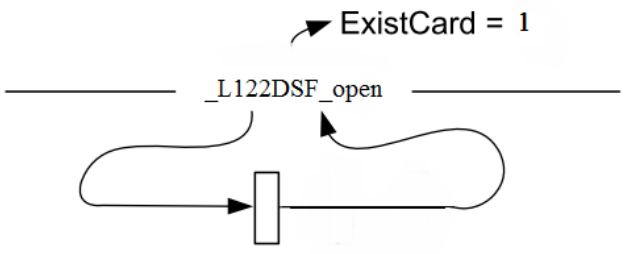

Figure 4-2: hardware initialization interface

4.1.2. Library Initialization

Motionnet library can be initialized by hardware device driver library call. With the Linkage between

hardware and function library, user can use different types of communication masters by the same software interface.

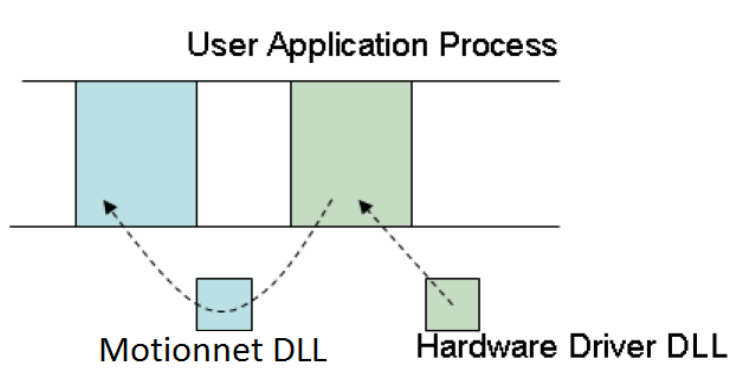

Figure 4-3: library relationship

The hardware needs to be initialized by calling the following software function calls.

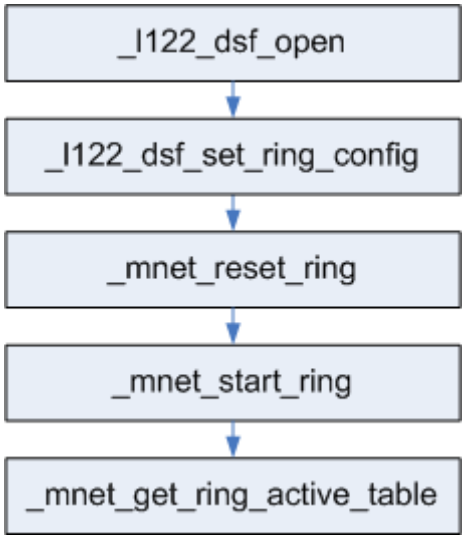

Figure 4-4: software function call sequence to initialize hardware

4.1.3. Motionnet Master

The operation of Motionnet extension is divided into the following 2 groups. One is the Motionnet master device, the other is the slave device.

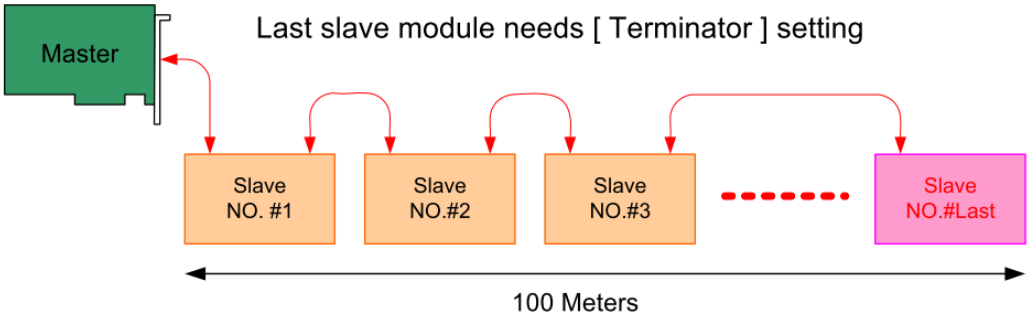

Figure 4-5: illustration of master-slave

4.2. Data Definition

Name	Description	Range
U8	8-bit ASCII character	0 to 255
I16	16-bit signed integer	-32768 to 32767
U16	16-bit unsigned integer	0 to 65535
I32	32-bit signed long integer	-2147483648 to 2147483647
U32	32-bit unsigned long integer	0 to 4294967295
F32	32-bit single-precision floating-point	-3.402823E38 to 3.402823E38
F64	64-bit double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic value	TRUE, FALSE

4.3. PCI-L122-DSF API List

Function Name	Description
Card resource functions	
_l122_dsf_open	Initialize hardware and resources.
_l122_dsf_close	Release hardware resources.
_l122_dsf_check_switch_card_num	Check the existence of the PCI-L122-DSF with a card number.
_l122_dsf_get_start_ring_num	Get the amount of the Motionnet Rings of the specified card.
_l122_dsf_set_ring_config	Set the baud rate of Motionnet Rings in the specified card.
Local digital I/O	
_l122_dsf_read_local_input	Obtain the input ON/OFF status of 8 local digital input channels which are in the CN3 port.
_l122_dsf_read_local_output	Obtain the output ON/OFF status of 8 local digital output channels which are in the CN3 port.
_l122_dsf_write_local_output	Set the 8 channels output data to the CN3 output port.
FRAM relation functions	
_l122_dsf_read_fram_byte	Read a byte data from the FRAM.
_l122_dsf_read_fram_word	Read a word data from the FRAM.
_l122_dsf_read_fram_dword	Read a double word data from the FRAM.
_l122_dsf_write_fram_byte	Write a byte data to the FRAM.
_l122_dsf_write_fram_word	Write a word data to the FRAM.
_l122_dsf_write_fram_dword	Write a double word data to the FRAM.
AES relation functions	
_l122_dsf_get_secure_id	Get the secure id from the security ASIC on PCI-L122-DSF board.
_l122_dsf_gen_aes_key	Generate AES key by inputting SI key using Rijndael algorithm. This key plays the role of “password”.
_l122_dsf_check_aes_key	Check the validity of the calculated AES key.

4.3.1. _l122_dsf_open

Format

I16 status = _l122_dsf_open (U16* existcards)

Purpose

Initialize hardware and resources.

Parameters

Name	Type	Description
existcards	U16 *	Get master card count in your PC

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.2. _l122_dsf_close

Format

I16 status = _l122_dsf_close ()

Purpose

Release hardware resources.

Parameters

N/A

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.3. _l122_dsf_check_switch_card_num

Format

I16 status = _l122_dsf_check_switch_card_num (U16 SwitchCardNo, U8 *IsExist)

Purpose

Check the existence of the PCI-L122-DSF with a card number.

Parameters

Name	Type	Description
SwitchCardNo	U16	The number of the card to be checked with the rotary switch setting.
IsExist	U8 *	Equal to 1 if the card exists, 0 if the card does not exist.

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.4. _l122_dsf_get_start_ring_num

Format

I16 status = _l122_dsf_get_start_ring_num (U16 SwitchCardNo, U16 *RingNo)

Purpose

Get the amount of the Motionnet Rings of the specified card.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
RingNo	U16 *	The amount of the Motionnet Rings.

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.5. _l122_dsf_set_ring_config

Format

I16 status = _l122_dsf_set_ring_config (U16 SwitchCardNo, U16 RingOfCard, U8 BaudRate)

Purpose

Set the baud rate of Motionnet Rings in the specified card.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
RingOfCard	U16	Ring Number 0 ~1
BaudRate	U8	Baud rate settings. Argument: 0 — Baud rate: 2.5Mbps Argument: 1 — Baud rate: 5Mbps Argument: 2 — Baud rate: 10Mbps Argument: 3 — Baud rate: 20Mbps

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.6. _l122_dsf_read_local_input

Format

I16 status = _l122_dsf_read_local_input (U16 SwitchCardNo, U8* Val)

Purpose

Obtain the input ON/OFF status of 8 local digital input channels which are in the CN3 port.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Val	U8 *	8 channel entrance point ON/OFF condition feedback value (0x00 ~0xff)

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

Note. Bit0~Bit7 represents the status of the GPIO in Channel #0~#7

Bit**0** = 1 => Ch. #0 is ON

Bit**0** = 0 => Ch. #0 is OFF

4.3.7. _l122_dsf_read_local_output

Format

I16 status = _l122_dsf_read_local_output (U16 SwitchCardNo, U8* Val)

Purpose

Obtain the output ON/OFF status of 8 local digital output channels which are in the CN3 port.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Val	U8 *	8 channel entrance point ON/OFF condition feedback value (0x00 ~0xff)

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

Note. Bit0~Bit7 represents the status of the GPIO in Channel #0~#7

Bit**0** = 1 => Ch. #0 is ON

Bit**0** = 0 => Ch. #0 is OFF

4.3.8. _l122_dsf_write_local_output

Format

I16 status = _l122_dsf_write_local_output (U16 SwitchCardNo, U8 Val)

Purpose

Set the 8 channels output data to the CN3 output port.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Val	U8	Output value (0x00 ~0xff) to be set to the output port.

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.9. _l122_dsf_read_fram_byte

Format

I16 status = _l122_dsf_read_fram_byte (U16 SwitchCardNo, U16 Offset, U8 *Val)

Purpose

Read a byte data from the FRAM.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Offset	U16	The offset related to the accessible FRAM address.
Val	U8 *	Byte value in the FRAM with designated offset.

Note. The offset must be within the range 0~0xffff

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.10. _l122_dsf_read_fram_word

Format

I16 status = _l122_dsf_read_fram_word (U16 SwitchCardNo, U16 Offset, U16 *Val)

Purpose

Read a word data from the FRAM.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Offset	U16	The offset related to the accessible FRAM address.
Val	U16 *	Word value in the FRAM with designated offset.

Note. The offset must be within the range 0~0xffff

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.11. _l122_dsf_read_fram_dword

Format

I16 status = _l122_dsf_read_fram_dword (U16 SwitchCardNo, U16 Offset, U32 *Val)

Purpose

Read a double word data from the FRAM.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Offset	U16	The offset related to the accessible FRAM address.
Val	U32 *	Double word value in the FRAM with designated offset.

Note. The offset must be within the range 0~0xffff

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.12. _l122_dsf_write_fram_byte

Format

I16 status = _l122_dsf_write_fram_byte (U16 SwitchCardNo, U16 Offset, U8 Val)

Purpose

Write a byte data to the FRAM.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Offset	U16	The offset related to the accessible FRAM address.
Val	U8	Byte value to be written to the FRAM.

Note. The offset must be within the range 0~0xffff

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.13. _l122_dsf_write_fram_word

Format

I16 status = _l122_dsf_write_fram_byte (U16 SwitchCardNo, U16 Offset, U16 Val)

Purpose

Write a word data to the FRAM.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Offset	U16	The offset related to the accessible FRAM address.
Val	U16	Word value to be written to the FRAM.

Note. The offset must be within the range 0~0xffff

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.14. _l122_dsf_write_fram_dword

Format

I16 status = _l122_dsf_write_fram_dword (U16 SwitchCardNo, U16 Offset, U32 Val)

Purpose

Write a double word data to the FRAM.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
Offset	U16	The offset related to the accessible FRAM address.
Val	U32	Double word value to be written to the FRAM.

Note. The offset must be within the range 0~0xffff

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.15. _l122_dsf_get_secure_id

Format

I16 status = _l122_dsf_get_secure_id (U16 SwitchCardNo, U8 SecureID[8])

Purpose

Get the secure id from the security ASIC on PCI-L122-DSF board.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
SecureID[8]	U8	The pointer to an 8-byte array indicating Secure ID.

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.16. _l122_dsf_gen_aes_key

Format

I16 status = _l122_dsf_gen_aes_key (U16 SwitchCardNo, U8 SI_Key[16], U8 SecureID[8], U8 AES_Key[16])

Purpose

Generate AES key by inputting SI key using Rijndael algorithm. This key plays the role of “password”.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
SI_Key [8]	U8	Pointer to a 16-element byte-array indicating SI key, every element is an integer ranged from 0 to 9.
SecureID[8]	U8	Pointer to a 8-element byte-array indicating Secure ID, if SecureID is NULL, the data read from Security ASIC will be used to generate AES_Key.
AES_Key[16]	U8	Pointer to a 16-element byte-array indicating AES key.

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

4.3.17. _l122_dsf_check_aes_key

Format

I16 status = _l122_dsf_check_aes_key (U16 SwitchCardNo, U8 SI_Key[16], U8 AES_Key[16], U8* Validity)

Purpose

Check the validity of the calculated AES key.

Parameters

Name	Type	Description
SwitchCardNo	U16	The rotary switch set number of the PCI-L122-DSF.
SI_Key [8]	U8	Pointer to a 16-element byte-array indicating SI key, every element is an integer ranged from 0 to 9.
AES_Key[16]	U8	Pointer to a 16-element byte-array indicating AES key.
Validity	U8 *	Pointer to a variable that receives the validity of the AES_Key. 1: valid, 0: invalid.

Status Return

Function Name	Description
ERR_NoError	The function finished execution successfully.
Other	Please reference to the Appendix error table.

5. Appendix

5.1. Definition of Error Codes

ERR_NoError	0
ERR_InvalidSwitchCardNumber	-1
ERR_SwitchCardNumberRepeated	-2
ERR_OpenCardFailed	-3
ERR_CardNotExist	-4
ERR_InvalidParameter1	-5
ERR_InvalidParameter2	-6
ERR_InvalidParameter3	-7
ERR_InvalidParameter4	-8
ERR_InvalidParameter5	-9
ERR_InvalidParameter6	-10
ERR_InvalidParameter7	-11
ERR_InvalidParameter8	-12
ERR_InvalidParameter9	-13
ERR_GetSecureIdFailed	-14
ERR_GenAesKeyFailed	-15
ERR_OpenInterruptEventFailed	-16